intel

Development Solutions

ICE™-5100 Emulator Tutorial Guide

Order Number: 167414-003

ICE™-5100 EMULATOR TUTORIAL GUIDE

Order Number: 167414-003

REV.	REVISION HISTORY	DATE
-001	Original Issue.	7/86
-002	Revised to include a list of tutorial screens and an index.	8/86
-003	Typeset Version.	9/86

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

CAUTION

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. As temporarily permitted by regulation, it has not been tested for compliance with the limits for Class A Computing Devices pursuant to Subpart J of Part 15 of FCC rules, which are designed to provide reasonable protection against such interference. Operation of this equipment in a residential area is likely to cause interference in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation
Literature Sales
P.O. Box 58130
Santa Clara, CA 95052-8130

CONTENTS

GETTING STARTED

To get started with your ICE™-5100 emulator, do the following:

- Install the ICE-5100 emulator system hardware as directed in the *ICE-5100 Emulator Installation Supplement.*
- Install the ICE-5100 user probe as directed in the *ICE-5100/nnn User Probe Supplement.*
- Run the on-line ICE-5100 emulator tutorial as explained in this supplement.

The *ICE-5100 Emulator Reference Manual* has the following structure.

Chapter 1	presents an overview of the ICE™-5100 emulator software.
Chapter 2	presents debugging techniques and advanced ICE-5100 emulator features.
Chapter 3	is an encyclopedia of ICE-5100 emulator commands, keywords, and related topics.
Appendix A	describes the state of the ICE-5100 emulator when power is first turned on.
Appendix B	contains a list of miscellaneous topics you should be aware of when using the ICE-5100 emulator.
Appendix C	describes use of the clips assembly and the hardware specifications on the clips assembly.
Appendix D	contains hardware specifications on the power supply and serial cable pin-outs.
Appendix E	lists the error messages displayed by the ICE-5100 emulator.
Appendix F	lists ASCII codes and their functions.
Appendix G	lists related reference publications.
Glossary	contains a list of terms used in the manual.
Index	contains an index of terms used in the manual.
Inside back cover	provides service information.

Other manuals for the ICE-5100 emulator include:

Installation Supplement	explains hardware and software installation, including information on confidence tests and how to make connections for interrupt measurements. Provides information on ICE-5100 emulator limitations and suggestions for use.
User Probe Supplement	Explains user probe specific hardware setup. There is a user probe supplement for each user probe.
Pocket Reference	contains a ready-reference to emulator commands, probe specific commands, keywords, and registers. There is a pocket reference for each user probe.

T

TUTORIAL GUIDE

intel

To quickly learn how to use the ICE™-5100 emulator commands and features, Intel recommends that you complete the on-line tutorial before proceeding to debug your own programs.

This guide supplements the on-line ICE-5100 emulator tutorial and is divided into four sections:

- Invocation — Section T.1 explains how to invoke the ICE-5100 emulator tutorial.
- Organization — Section T.2 explains how the ICE-5100 emulator tutorial is organized and how to use the tutorial. It also lists all the tutorial screens.
- Tutorial index — Section T.3 contains an index of topics discussed in the on-line tutorial.
- Program listing — Section T.4 contains a list file of the PL/M-51 MESSG program used in the tutorial. The list file includes a PL/M-51 listing and an equivalent ASM-51 listing.

T.1 Tutorial Invocation

The ICE-5100 emulator tutorial is easy to access.

1. Prepare your host to use the ICE-5100 emulator tutorial. Refer to the *ICE™-5100 Installation Supplement*, order number 167095, for instructions on installing the ICE-5100 emulator software and tutorial files on your host computer.

   - IBM PC AT and PC XT users must change the directory to the directory containing the tutorial files. For example, if your tutorial files are in the directory named TUTDIR:

     ```
     CD C:\TUTDIR<Enter>
     ```

   - Intel Series IV users must assign a logical device to the directory containing the tutorial files. For example:

     ```
     ASSIGN :F0: TO /WD0/TUTDIR<RETURN>
     ```

   - Intel Series III users on a network must assign a logical device to the directory containing the tutorial files, and another logical device to the ICE-5100 emulator software. For example:

     ```
     ASSIGN :F0: TO /WD0/TUTDIR<RETURN>
     ASSIGN :F1: TO /WD0/ICEDIR<RETURN>
     ```

   - Intel Series III stand-alone users must place the tutorial diskette in drive 0 and the ICE-5100 emulator diskette in drive 1 of the host computer system.

NOTE

The ICE-5100 emulator must be in stand-alone mode for the tutorial to execute (refer to Figure T-1).

2. Invoke the emulator software. For example, assuming your ICE-5100 emulator software is in a directory named ICEDIR (where *nnn* is the number of your user probe, e.g., ICE252):

   IBM hosts: `\ICEDIR\ICEnnn<Enter>`

   Intel hosts:

   Series III: `RUN :F1:ICEnnn<RETURN>`

   Series IV: `@DO/ICEDIR/ICEnnn<RETURN>`

3. You can activate the tutorial whenever you have the ICE-5100 emulator hlt> prompt. Enter the following command from the hlt> prompt.

   - For IBM PC AT and PC XT hosts, enter:

     `hlt>INCLUDE TUTOR NOLIST<Enter>`

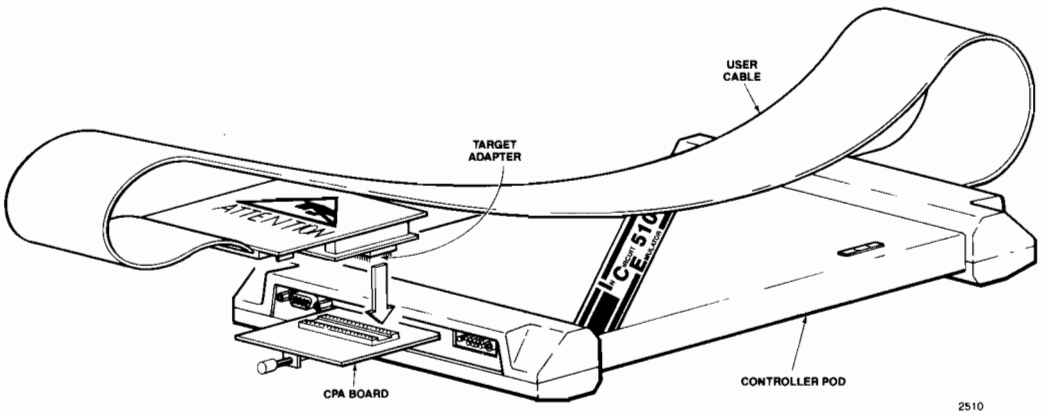

Figure T-1 ICE™-5100 Emulator in Stand-alone Mode

- For Intel Series III and Series IV hosts, enter:

```
hlt>INCLUDE :F1:TUTOR NOLIST<RETURN>
```

The following message is displayed:

```
**************************************************
*                                                *
*      WELCOME TO THE ICE-5100 EMULATOR TUTORIAL      *
*                                                *
**************************************************
NOTE: This tutorial will RESET the ICE-5100 emulator.
Do you wish to continue? (Y or N)<Enter>
```

If you do not want your ICE-5100 emulator RESET (which will clear all pre-set conditions), or you do not wish to continue with the tutorial, enter N. Otherwise, enter Y to continue the initialization process. Figure T-2 shows the first tutorial screen.

```
                                          SCR1: WELCOME TO ICE
     ICE-5100 EMULATOR TUTORIAL (DOS Version 1.0)   N    = Next screen
           Copyright 1986 Intel Corporation         R    = Rewrite SCR1
Welcome to the ICE-5100 tutorial. This tutorial     PR   = Previous screen
will teach you how to use the ICE-5100 emulator.    M    = Go to main menu
                                                    Q    = Quit tutorial
Note the box to the right. This box appears in      SCR# = Screen desired
each screen. It provides the name and title of
the current screen and shows which keys to enter to move to other tutorial
screens or to exit from the tutorial. (To jump to a specific screen, enter
"SCR#" where # is the number of the screen desired.)

The ICE-5100 emulator prompt ( hlt> ) appears at the bottom of the text under
a horizontal line. Enter commands from this prompt in either uppercase or
lowercase letters. Use the <Rubout> key ( <-- at the top of the keyboard) to
correct a command. Press the <Enter> key to execute the command.

----------------Enter N <Enter> to continue with the tutorial----------------
hlt>
```

Figure T-2 Tutorial Introductory Screen: SCR1

T.2 Tutorial Organization

The tutorial is divided into a main path and a set of feature modules (refer to Figure T-3 for an overview of tutorial organization). The main path is divided into two modules. The first main-path module (MOD1) guides you through defining the debug environment and running a sample program that has a bug in it. The second main-path module (MOD2) guides you through finding and fixing the program bug. The feature modules elaborate on topics mentioned in the main path (refer to Figure T-4).

Each screen and module has a name (e.g., SCR5, SCRC3, MODC, FMOD). Typing a screen name causes that screen to be displayed. Typing a module name sets up any prerequisites needed to carry out the steps in that module, and then displays the first screen of the module.

The ICE-5100 emulator tutorial screens are created with ICE-5100 emulator commands. When you use the tutorial, you are also using ICE-5100 emulator software. As a consequence, whenever the hlt> appears, you can enter any ICE-5100 emulator commands you wish.

```
The purpose of this tutorial is to help you learn
the ICE-5100 emulator command language and to
demonstrate a debugging session.

The tutorial is organized into modules. A module
is a sequence of screens of information and examples.
There are two groups of modules: MAIN PATH modules
(debugging skills), and FEATURE modules (supple-
mentary information on main path topics). The following modules are available:

 MOD1  Main Path: Basic debugging skills
 MOD2  Main Path: Advanced debugging skills
 FMOD  Features:  Information on specific emulator features

All commands are executed by pressing the <Enter> key after the command name.
Select MOD1, MOD2, or FMOD now.

--------------------------------------------------------------------------------
hlt>
```

```
SCR2: MAIN MENU
N    = Next screen
R    = Rewrite SCR1
PR   = Previous screen
M    = Go to main menu
Q    = Quit tutorial
SCR#= Screen desired
```

Figure T-3 Tutorial Main Menu: SCR2

Once you enter the commands recommended on a particular tutorial screen, you need not immediately advance to the next screen. Instead, you are encouraged to experiment with commands to ensure that you understand the concepts presented.

When you are ready for the next screen, you can call it by typing N followed by <Enter> (or <RETURN>).

If the screen scrolls out of view before you are finished, it can be redisplayed by typing R followed by <Enter> (or <RETURN>).

T.2.1 Copies of Tutorial Screens

For your convenience, the two ICE-5100 emulator tutorial (DOS version) menu screens are shown in the following figures:

Figure T-3 Tutorial Main Menu: SCR2
Figure T-4 Menu of Features Modules: FMOD

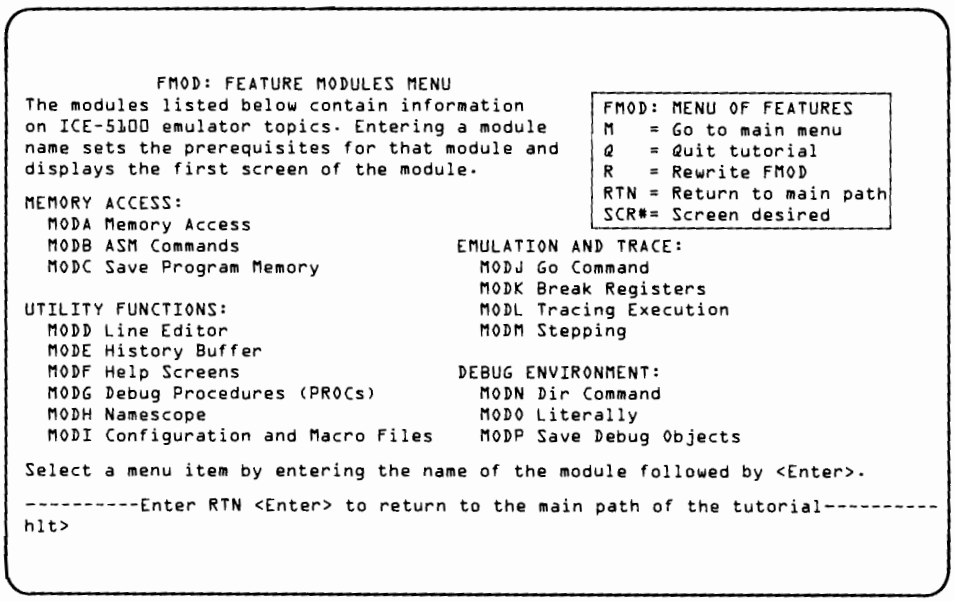

Figure T-4 Tutorial Feature Modules Menu: FMOD

The tutorial main menu gives an overview of the organization of the tutorial. To display this menu on your screen, enter:

```
hlt>M <Enter> (or <RETURN>)
```

Some topics are briefly introduced in the main tutorial path and are explained in more detail in the feature modules. Figure T-4 shows the menu for the feature modules. To display this menu on your screen, enter the following:

```
hlt>FMOD<Enter> (or <RETURN>)
```

T.2.2 List of All Tutorial Screens

Tables T-1 and T-2 list all of the tutorial screens, as follows:

Table T-1 Main Path Screens
Table T-2 Feature Module Screens

Each module is a major division of the tutorial. The modules are entered by typing the name of the module (e.g. MOD2, MODG, MODK).

Table T-1 Main-Path Screens

Module Name	Screen Name	Screen Title	Topic
	SCR1	Welcome To ICE	These screens are discussed in Section T.2.1.
	SCR2	Main Menu	
	FMOD	Menu of Features	
MOD1	SCR3	Introduction	BASIC DEBUGGING SKILLS This module describes mapping memory, loading a program, listing and accessing program symbols, setting NAMESCOPE, and executing a program.
	SCR4	Intro to LOAD	
	SCR5	Memory Mapping 1	
	SCR6	Memory Mapping 2	
	SCR7	LOAD MESSG	
	SCR8	DIR	
	SCR9	Symbol Access 1	
	SCR10	Symbol Access 2	
	SCR11	NAMESCOPE	
	SCR12	Symbol Access 3	
	SCR13	GO	
	SCR14	End of MOD 1	
MOD2	SCR15	Module 2	ADVANCED DEBUGGING SKILLS This module describes creating a debug procedure (PROC), defining break registers (BRKREG), verifying program variables, displaying and changing program ASM code, tracing program execution, stepping through program execution, and saving program patches and debug objects.
	SCR16	PROCs 1	
	SCR17	PROCs 2	
	SCR18	Break Registers	
	SCR19	GO USING BRKREG	
	SCR20	Verify Temp 1	
	SCR21	Verify Temp 2	
	SCR22	Showvar	
	SCR23	GO USING	
	SCR24	ASM 1	
	SCR25	ASM 2	
	SCR26	Verify patch	
	SCR27	Verify Program	
	SCR28	TRACE	
	SCR29	Stepping	
	SCR30	SAVE and PUT	
	SCR31	Setting Environ	
	SCR32	End of Tutorial	

Table T-2 Feature Module Screens

Module Name	Screen Name	Screen Title	Topic
MODA	SCRA1 SCRA2 SCRA3 SCRA4 SCRA5	Memory Access 1 Memory Access 2 Memory Access 3 Memory Access 4 Memory Access 5	MEMORY ACCESS This module demonstates the use of MTYPE commands to display and modify memory. It also describes the BASE command.
MODB	SCRB1 SCRB2 SCRB3	ASM Commands 1 ASM Commands 2 ASM Commands 3	ASM COMMANDS This module demonstrates using the ASM command to change program code.
MODC	SCRC1	SAVE Memory	SAVE PROGRAM MEMORY This module demonstrates copying program memory to a directory file.
MODD	SCRD1	Line Editor	LINE EDITOR This module demonstrates how to modify command strings.
MODE	SCRE1	History Buffer	HISTORY BUFFER This module demonstrates the use of the command history buffer.
MODF	SCRF1 SCRF2	HELP Screens 1 HELP Screens 2	HELP SCREENS This module demonstrates the use of the HELP screens.
MODG	SCRG1 SCRG2 SCRG3 SCRG4 SCRG5 SCRG6 SCRG7	Debug PROCs 1 Debug PROCs 2 Debug PROCs 3 Debug PROCs 4 Debug PROCs 5 Debug PROCs 6 Debug PROCs 7	DEBUG PROCEDURES This module demonstrates how to create various types of debug procedures using various compound-command constructs.
MODH	SCRH1 SCRH2 SCRH3 SCRH4 SCRH5	NAMESCOPE 1 NAMESCOPE 2 NAMESCOPE 3 NAMESCOPE 4 NAMESCOPE 5	NAMESCOPE This module demonstrates the use of NAMESCOPE to reduce the amount of information needed to reference user program symbols.
MODI	SCRI1 SCRI2	Macro Files 1 Macro Files 2	CONFIGURATION AND MACRO FILES This module demonstrates the use of configuration and macro files to automatically initialize the system.
MODJ	SCRJ1 SCRJ2 SCRJ3 SCRJ4	GO Command 1 GO Command 2 GO Command 3 GO Command 4	GO COMMAND This module demonstrates the use of the GO command to begin and control emulation.

Table T-2 Feature Module Screens (continued)

Module Name	Screen Name	Screen Title	Topic
MODK	SCRK1 SCRK2 SCRK3 SCRK4 SCRK5	Break Registers 1 Break Registers 2 Break Registers 3 Break Registers 4 Break Registers 5	BREAK REGISTERS This module demonstrates the use of break registers (BRKREGs) to control program execution.
MODL	SCRL1 SCRL2 SCRL3 SCRL4 SCRL5	Trace 1 Trace 2 Trace 3 Trace 4 Trace 5	TRACING EXECUTION This module demonstrates the use of the trace buffer and trace registers (TRCREGs) to monitor program execution.
MODM	SCRM1 SCRM2 SCRM3	Stepping 1 Stepping 2 Stepping 3	STEPPING This module demonstrates the use of the ISTEP and LSTEP commands to step through program execution.
MODN	SCRN1 SCRN2	DIR Command 1 DIR Command 2	DIR COMMAND This module demonstrates the use of the DIR command to display program symbols and debug object names.
MODO	SCRO1 SCRO2	LITERALLY 1 LITERALLY 2	LITERALLY This module demonstrates the use of the LITERALLY command to abbreviate character strings, commands, and emulator keywords.
MODP	SCRP1 SCRP2 SCRP3 SCRP4 SCRP5	Save Debug 1 Save Debug 2 Save Debug 3 Save Debug 4 Save Debug 5	SAVE DEBUG OBJECTS This module demonstrates the use of the PUT, APPEND, and INCLUDE commands to save and retrieve debug objects from a directory file.

T.3 Tutorial Index

The following index correlates ICE-5100 emulator tutorial module and screen names with tutorial topics and emulator commands. To display any module or screen cited in the index, enter the module or screen name followed by <Enter> (or <RETURN>). For example, to display the BASE command information in screen SCRA2, enter SCRA2 <Enter> (or <RETURN>).

Subject	Module / Screen Name
EA pin	SCR5
Error messages	SCRF2
Esc key	SCR8
Execution, program	MODJ
EXIT	SCR14, SCR32
FOREVER	SCRJ3
FROM	SCR19, SCRJ2
FROM 0	SCR27
Fully qualified reference	SCR10
GO	MODJ
GO FOREVER	SCRJ3
GO FROM	SCR19, SCRJ2
GO TIL	SCRJ2
GO USING	SCR19
HALT	MODJ
HELP	MODF
Hexadecimal	SCRA2
History buffer	MODE
IDATA	MODA
IF	SCRG3
INCLUDE	SCRP3, SCRP4, SCRP5
ISTEP	SCR29, MODM
Keywords	SCR12
Label, access	SCR12, MODA
LENGTH	MODA
Line editor	MODD
Line numbers	SCR20
LITERALLY	MODO
LOAD	SCR7, MODC
LSTEP	SCR29, MODM
Macro file	SCRI2
MAP	SCR5
Memory access	MODA
Memory address spaces	SCRA5
Menu, syntax	SCRG
Menu, tutorial modules	SCR2, FMOD
Mtype (memory type)	MODA
NAMESCOPE	MODH
Nesting	SCRG2
NEWEST	SCRL2
Number base	SCRA2

Subject	Module / Screen Name
OUTSIDE	SCRJ3
Patch, program	SCR24, MODB
PRINT	SCR28, MODL
PROC	SCR16, MODG
Procedures, debug	MODG
Program counter ($)	SCRL1
PUT	SCR30, SCRP1, SCRK4
RDATA	SCRA5
REMOVE	SCRG8, SCRK5
REPEAT	SCRG5
RETURN	SCR17, SCRK2
SAVE	SCR30, MODC
Screen control	SCR16
Set debug environment	SCR31, MODI
Symbol	SCR9, MODA
Symbol, address of	SCR12
Symbol buffer size	SCRI1
Syntax menu	SCR6
Tab key	SCR6, SCRN2
TIL	SCRJ2
TO	SCR24, SCRA1
TRACE	SCR28, MODL
Trace register (TRCREG)	SCRL3
UNTIL	SCRG5
USING	SCR19
Variable, address of	SCR12, MODA
Variable, debug	SCRG3
Variable, value of	SCR9, MODA
WHILE	SCRG5
WRITE	MODD, SCR22
XDATA	SCRA5

T.4 Tutorial Program Listings

The sample program (MESSG) used in the tutorial is written in PL/M-51. The program displays the following message:

```
Intel ICE-5100 Emulators are HOT-
```

The display is rotated by shifting the leftmost character to the right most location. The rotation is triggered by a timer interrupt.

There are two files on the tutorial disk associated with the MESSG program:

MESSG	compiled, linked, and located absolute code
MESSG.LST	program listings in PL/M-51 code and ASM code

NOTE

The absolute addresses in the ASM program listing have been modified to match the actual program addresses when the program is loaded at location 0H.

The tutorial frequently references the PL/M-51 program listing.

T.4.1 Program Listing for the MESSG Program

The following PL/M-51 listing of the MESSG program contains a bug on line #28 that is found and corrected in the tutorial.

```
DOS 3.10 (036-N) PL/M-51 V1.2
COMPILER INVOKED BY: C: DOSLANG PLM51.EXE MESSG.PLM DEBUG SYMBOLS CODE
      /*************************************************************************/
      /*                                                                       */
      /*                    MESSG.PLM (with bug in line #28)                   */
      /*    PL/M-51 program which rotates a character string in a buffer       */
      /*    according to a timer. This program does not contain any            */
      /*    target hardware specific procedures.                               */
      /*                                                                       */
      /*    This program is used as the sample program for the ICE-5100        */
      /*    emulator tutorial.                                                 */
      /*                                                                       */
      /*************************************************************************/

 1  1  MAIN_DISPLAY: DO;

      /********************** VARIABLE DECLARATIONS **********************/

 2  1  DECLARE  forever                LITERALLY              'WHILE 1',
               false                  LITERALLY              '0',
               true                   LITERALLY              'NOT false',

               /*User variables */
```

```
                reset_low               BYTE CONSTANT (083H),
                reset_high              BYTE CONSTANT (0E8H),

                disp_buffer (50)        BYTE,
                buff_size               BYTE,
                int_flag                BIT,
                message (*)             BYTE CONSTANT
                (' Intel ICE-5100 Emulators are HOT-'),
                i                       BYTE,

                /* Timer 0 processor specific locations */

                tm0_low                 BYTE at (08AH) REGISTER,
                tm0_high                BYTE at (08CH) REGISTER;

        /************** INITIALIZATION PROCEDURE DECLARATION **************/

 3  2  INIT: PROCEDURE;                /* Procedure to initialize timers */

                            /* Timer 0 related special function registers */

 4  2    DECLARE  TMOD                          BYTE at (089H) REGISTER,
                  ET0                           BIT at (0A9H) REGISTER,
                  EA                            BIT at (0AFH) REGISTER,
                  TR0                           BIT at (08CH) REGISTER;

        /* Beginning of initialization code */

 5  2    buff_size = LAST(message);          /* Store the maximum index value */

 6  3    DO i = 0 TO (buff_size);
 7  3     disp_buffer(i) = message(i);        /* Fill the display buffer */
 8  3    END;

 9  2    tm0_low  = reset_low;                    /* Initialize timer 0 */
10  2    tm0_high = reset high;
11  2    TMOD     = 01H;                 /* Timer 0 in 16 bit timer mode */
12  2    ET0      = 1;                   /* Set timer 0 interrupt enable */
13  2    EA       = 1;                    /* Set global interrupt enable */
14  2    TR0      = 1;                    /* Set timer 0 run control bit */
15  2    int_flag = false;                  /* Initialize interrupt flag */

16  1  END INIT;

        /************** TIMER 0 INTERRUPT ROUTINE DECLARATION **************/

17  2  SERVICE: PROCEDURE INTERRUPT 1;

18  2    int_flag = true;                     /* Set the interrupt flag */
19  2    tm0_low = reset_low;                         /* reset timer 0 */
20  2    tm0_high = reset_high;
```

```
21  2  END;

       /************** PROCEDURE TO PRINT THE CHARACTER BUFFER **************/

22  2  CHAR_DISPLAY: PROCEDURE;

                 /* PROGRAM STUB: DISPLAY CHARACTERS ON TARGET HARDWARE */

23  2  END;

       /*************** PROCEDURE TO ROTATE CHARACTER BUFFER ***************/

24  2  ROTATE: PROCEDURE;

25  2    DECLARE  INDEX_PTR  BYTE;              /* Temporary variables */
                  TEMP       BYTE;

         /* Begin subroutine processing */

26  2    INDEX_PTR = 1;
27  2    TEMP      = disp buffer(0);
                                              /* Rotate all the characters */
28  3    DO WHILE (index_ptr < buff_size);       /* BUG: should be "<=" */

29  3      disp_buffer(index ptr - 1) = disp_buffer(index_ptr);
30  3      INDEX_PTR = INDEX_PTR + 1;

31  3    END;

32  2    disp_buffer(buff_size) = TEMP;         /* Put first char at end */

33  2  PRINT: CALL CHAR_DISPLAY; /* Display the string on the target hardware */

34  2    int_flag = false;                    /* Reset the interrupt flag */

35  2  END;

       /*************************************************************************/
       /*******************     BEGIN MAIN PROGRAM     ********************/
       /*************************************************************************/

36  1  START: CALL INIT;                              /* Initialize timer */

37  2      DO forever;                 /* Call rotate after the software */
                                          /* interrupt sets the flag */
38  2        IF (int_flag) THEN
39  2          CALL ROTATE;

40  2      END;

41  1  END MAIN_DISPLAY;
```

```
                ; PROCEDURE MAIN_DISPLAY (START)
                                        ; STATEMENT # 3
                ; PROCEDURE INIT (START)
                                        ; STATEMENT # 5
000E 755322     F           MOV     BUFF_SIZE,#22H
                                                ; STATEMENT # 6
0011 755400     F           MOV     I,#00H
0014             DO?1:
0014 E554       F           MOV     A,I
0016 D3                     SETB    C
0017 9553       F           SUBB    A,BUFF_SIZE
0019 5016                   JNC     DOEND?2
                                                ; STATEMENT # 7
001B E554       F           MOV     A,I
001D 900095     F           MOV     DPTR,#MESSAGE
0020 93                     MOVC    A,@A+DPTR
0021 FE                     MOV     R6,A
0022 E554       F           MOV     A,I
0024 2421       F           ADD     A,#DISP_BUFFER
0026 F8                     MOV     R0,A
0027 A606                   MOV     @R0,AR6
                                                ; STATEMENT # 8
0029 7854       F           MOV     R0,#I
002B 7401                   MOV     A,#01H
002D 26                     ADD     A,@R0
002E F6                     MOV     @R0,A
002F 50E3                   JNC     DO?1
0031            DOEND?2:
                                                ; STATEMENT # 9
0031 900093     F           MOV     DPTR,\RESET_LOW
0034 E4                     CLR     A
0035 93                     MOVC    A,@A+DPTR
0036 F58A                   MOV     TM0_LOW,A
                                                ; STATEMENT # 10
0038 900094     F           MOV     DPTR,\RESET HIGH
003B E4                     CLR     A
003C 93                     MOVC    A,@A+DPTR
003D F58C                   MOV     TM0_HIGH,A
                                                ; STATEMENT # 11
003F 758901                 MOV     TMOD,#01H
                                                ; STATEMENT # 12
0042 D2A9                   SETB    ET0
                                                ; STATEMENT # 13
0044 D2AF                   SETB    EA
                                                ; STATEMENT # 14
0046 D28C                   SETB    TR0
                                                ; STATEMENT # 15
0048 C200       F           CLR     INT_FLAG
```

```
                                        ; STATEMENT # 16
004A 22                 RET
                 ; PROCEDURE INIT (END)
                                        ; STATEMENT # 17
                 ; PROCEDURE SERVICE (START)
                                        ; STATEMENT # 18
004B D3                 SETB  C
004C 9200      F        MOV   INT_FLAG,C
                                        ; STATEMENT # 19
004E 900093    F        MOV   DPTR,#RESET_LOW
0051 E4                 CLR   A
0052 93                 MOVC  A,@A+DPTR
0053 F58A               MOV   TMO_LOW,A
                                        ; STATEMENT # 20
0055 900094    F        MOV   DPTR,#RESET_HIGH
0058 E4                 CLR   A
0059 93                 MOVC  A,@A+DPTR
005A F58C               MOV   TMO_HIGH,A
                                        ; STATEMENT # 21
005C 22                 RET
                 ; PROCEDURE SERVICE (END)

                                        ; STATEMENT # 22
                 ; PROCEDURE CHAR_DISPLAY (START)
                                        ; STATEMENT # 23
005D 22                 RET
                 ; PROCEDURE CHAR_DISPLAY (END)

                                        ; STATEMENT # 24
                 ; PROCEDURE ROTATE (START)
                                        ; STATEMENT # 26
005E 750801    F        MOV   INDEX_PTR,#01H
                                        ; STATEMENT # 27
0061 852109    F        MOV   TEMP,DISP_BUFFER
                                        ; STATEMENT # 28
0064            WHILE?5:
0064 E508      F        MOV   A,INDEX_PTR
0066 C3                 CLR   C
0066 9553      F        SUBB  A,BUFF_SIZE
0069 13                 JNC   WEND?6
                                        STATEMENT # 29
006B E508      F        MOV   A,INDEX_PTR
006D 2421      F        ADD   A,#DISP_BUFFER
006F F8                 MOV   R0,A
0070 E508      F        MOV   A,INDEX_PTR
0072 14        DEC      A
0073 8606               MOV   AR6,@R0
0075 2421      F        ADD   A,#DISP_BUFFER
```

```
0077 F8                 MOV   R0,A
0078 A606               MOV   @R0,AR6
                                      ; STATEMENT # 30

007A 0508       F       INC   INDEX_PTR
                                      ; STATEMENT # 31
007C 80E6               SJMP WHILE?5
007E            WEND?6:
                                      ; STATEMENT # 32
007E E553       F       MOV   A,BUFF_SIZE
0080 2421       F       ADD   A,#DISP_BUFFER
0082 F8                 MOV   R0,A
0083 A609       F       MOV   @R0,TEMP
                                      ; STATEMENT # 33
0085            PRINT:
0085 115D       F       ACALL CHAR_DISPLAY
                                      ; STATEMENT # 34
0087 C200       F       CLR   INT_FLAG
                                      ; STATEMENT # 35
0089 22                 RET
                 ; PROCEDURE ROTATE (END)

                                      ; STATEMENT # 36
008A            START:
008A 110E       F       ACALL INIT
                                      ; STATEMENT # 37
008C            WHILE?7:
                                      ; STATEMENT # 38
008C 300002     F       JNB   INT_FLAG,THEN?9
                                      ; STATEMENT # 39
008F 115E       F       ACALL ROTATE
                                      ; STATEMENT # 40
0091            THEN?9:
0091 80F9               SJMP  WHILE?7
0093            WEND?8:
                                      ; STATEMENT # 41
                ; PROCEDURE MAIN DISPLAY (END)
```

```
DEFN  SPACE  SIZE  NAME           ATTRIBUTES
  2   DATA     1   BUFF_SIZE.....BYTE
 22   CODE     1   CHAR_DISPLAY..PROCEDURE USING(0) STACK=02H
  2   DATA    50   DISP_BUFFER...BYTE ARRAY(50)
  4                EA............BIT REGISTER AT(AFH)
  4                ET0...........BIT REGISTER AT(A9H)
  2                FALSE.........LITERALLY
  2                FOREVER.......LITERALLY
  2   DATA     1   I.............BYTE
 25   DATA     1   INDEX_PTR.....BYTE
  3   CODE    61   INIT..........PROCEDURE USING(0) STACK=02H
  2   BIT      1   INT_FLAG......BIT
                   LAST..........BUILTIN
  1   CODE     9   MAIN_DISPLAY..MODULE
  2   CODE    35   MESSAGE.......BYTE ARRAY(35)
 33   CODE         PRINT.........LABEL
  2   CODE     1   RESET_HIGH....BYTE
  2   CODE     1   RESET_LOW.....BYTE
 24   CODE    44   ROTATE........PROCEDURE USING(0) STACK=02H
 17   CODE    18   SERVICE.......PROCEDURE USING(0) STACK=09H INTERRUPT(1)
 36   CODE         START.........LABEL
 25   DATA     1   TEMP..........BYTE
  2                TMO_HIGH .....BYTE REGISTER AT(8CH)
  2                TMO_LOW ......BYTE REGISTER AT(8AH)
  4                TMOD..........BYTE REGISTER AT(89H)
  4                TR0...........BIT REGISTER AT(8CH)
  2                TRUE..........LITERALLY

WARNINGS:
  1 IS THE HIGHEST USED INTERRUPT

MODULE INFORMATION:          (STATIC+OVERLAYABLE)

  CODE SIZE                = 0085H      133D
  CONSTANT SIZE            = 0025H      37D
  DIRECT VARIABLE SIZE     =   34H+02H  52D+  2D
  INDIRECT VARIABLE SIZE   =   00H+00H   0D+  0D
  BIT SIZE                 =   01H+00H   1D+  0D
  BIT-ADDRESSABLE SIZE     =   00H+00H   0D+  0D
  AUXILIARY VARIABLE SIZE  = 0000H       0D
  MAXIMUM STACK SIZE       = 0011H      17D
  REGISTER-BANK(S) USED:     0
  129 LINES READ
```